PRIVACY PRESERVING ENCRYPTED PHONETIC SEARCH OF SPEECH DATA

Cornelius Glackin[1], Gerard Chollet[1], Nazim Dugan[1], Nigel Cannings[1], Julie Wall[2], Shahzaib Tahir[3], Indranil Ghosh Ray[3], and Muttukrishnan Rajarajan[3]*

[1] Intelligent Voice Ltd., London, UK [2] University of East London, London, UK [3] City University London, London, UK
Email: neil.glackin@intelligentvoice.com[*]

ABSTRACT

This paper presents a strategy for enabling speech recognition to be performed in the cloud whilst preserving the privacy of users. The approach advocates a demarcation of responsibilities between the client and server-side components for performing the speech recognition task. On the client-side resides the acoustic model, which symbolically encodes the audio and encrypts the data before uploading to the server. The server-side then employs searchable encryption to enable the phonetic search of the speech content. Some preliminary results for speech encoding and searchable encryption are presented.

Index Terms— Speech recognition, privacy, searchable encryption, GPGPU computing

1. INTRODUCTION

In many activities involving big data, cloud computing offers a common distributed infrastructure for the storage of large amounts of data in a scalable, efficient, and low cost way. For sensitive data there is the possibility to use encryption for the secure storage of data in the cloud. However, whilst we have become increasingly good at encrypting data at rest, in order to process the data on the cloud we first need to decrypt it, which in turn excludes the possibility for using the cloud's resources to process sensitive data, unless it can be done in a secure way.

Speech contains biometric and other information which should remain private and therefore inaccessible to the cloud provider. Cloud users want to hide sensitive data such as speech, from cloud providers; similarly, companies using cloud services want to protect their intellectual property from cloud providers and users. Hence the need for strategies for processing data securely in the cloud becomes increasingly more important.

Speech is not reproducible in the sense that no speaker is capable of making the same utterance the same way twice, there are always small acoustic differences between utterances that have the same base transcription and this is a particular challenge of performing encrypted speech recognition. By reformulating the typical speech recognition task in such a way as to facilitate cloud computation, for example by reducing speech recognition to a search procedure [1], we demonstrate how secure speech processing in the cloud can be realized.

2. PROPOSED ARCHITECTURE

Our proposed solution involves the compression of speech containing biometric identifiers to a symbolic representation [1] that anonymizes the users' identity, and then on the other hand to use searchable symmetric encryption [2] to enable the finding of strings of symbols (e.g. phones) in an encrypted speech transcription. Encrypted string matching will then be performed to realize the language modelling component of the speech recognition system [3]. Fig. 1 illustrates the concept and the demarcation of responsibilities between the client and cloud server.

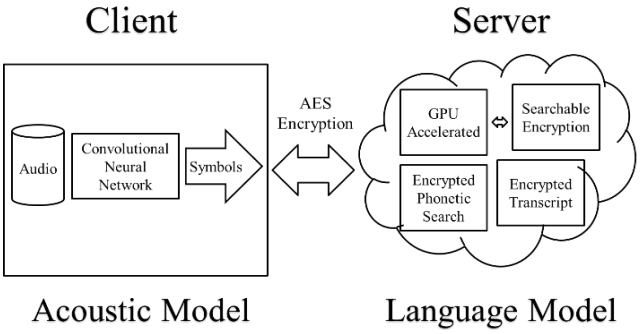

Fig. 1. The client and cloud server concept and division of responsibilities for the speech processing task

Speech recognition is typically broken down into acoustic and language modelling tasks. The acoustic model converts raw speech wave forms into acoustic units such as phones. The language model incorporates natural language processing and Bayesian probability theory to infer the text transcription, given what is known of a particular language, and what words the sequences of phones likely correspond to. As can be seen from Fig. 1, in the proposed system, the acoustic model resides on the client-side and the language model resides on the server/cloud-side.

3. PRIVACY PRESERVING SPEECH ENCODING

Regarding privacy preservation, in the cloud modality we need to make sure that personal information is not shared on the cloud. Since speech is a biometric data type, it is possible to identify someone and accurately infer a whole host of information that extends beyond the obvious information such as gender, to data such as height, weight, age, health and so on. Hence we need to ensure that speech itself is never in an unencrypted form on the cloud.

Traditionally, Automatic Speech Recognition (ASR) involves multiple successive layers of feature extraction to compress the amount of information processed from the raw audio so that the training of the acoustic model does not take an unreasonably long time. However, in recent years with increases in computational speed, adoption of parallel computation with GPGPUs, and advances in neural networks, many researchers are replacing traditional ASR algorithms with data-driven approaches that simply take the audio data in its frequency form (e.g. spectrogram) and process it with a Deep Neural Network (DNN), or more appropriately (since speech is temporal) with a Recurrent Neural Network (RNN) that can be trained quickly with GPGPUs. The RNN then converts the spectrogram directly to phonetic symbols and in some cases directly to text [4].

The problem with many of these approaches from the encryption point of view is that they typically combine the acoustic model and the language model with one neural network. This involves aligning the acoustic data (containing sensitive biometrics) at various stages of the network training with the text transcription with Expectation Maximization, Viterbi Search or Connectionist Temporal Classification [5]. In our approach, we propose that a higher level of privacy preservation can be attained by separating the acoustic and language model training between the client and server-sides of the system. Thus we need a way to train the acoustic model in isolation to the language model. In the acoustic model we use spectrograms as input and phonemes as output classes for training with a Convolutional Neural Network (CNN). Being able to train a system to identify time-frequency intervals in acoustic data and relate it to acoustic units such as phonemes requires extremely accurate labelling of acoustic data, and this is afforded by the well-known TIMIT speech corpus [6].

3.3. Preliminary Experiments

We use an implementation of the GoogLeNet architecture [7, 8] with Stochastic Gradient Descent (SGD) for training with the phonetic transcription within the TIMIT corpus. Once the CNN acoustic model is trained it is then uploaded to the client-side and used to perform inferencing, encoding the audio as phonetic symbols. The audio is first converted to Short-Term Fourier Transform (STFT) spectrograms and passed to the trained CNN which classifies the sliding windows operating over the spectrogram into phonetic symbols. Fig. 2 illustrates the operation of this convolutional speech encoder, where the sliding windows operating over the spectrogram (each one is 256×256 greyscale pixels) are classified by the CNN into phoneme classes. These are then encrypted with AES and uploaded to the cloud. Hence as well as storing encrypted audio the cloud also stores a symbolic representation of the encrypted speech data.

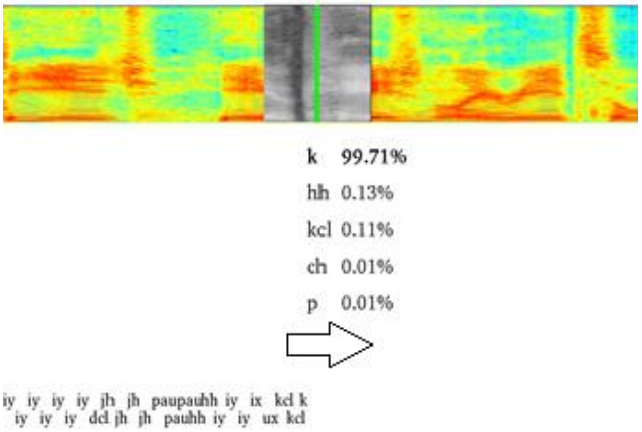

Fig. 2. Client-side convolutional speech encoder

4. ENCRYPTED PHONETIC SEARCH

We make some assumptions regarding the potential places of attacks that we need to address for the development of the security of the speech processing system. The server is assumed to be 'friendly but curious', meaning that it will not actively try and break the encryption but will monitor the traffic and infer the content of the audio if it can, which means that the data should be encrypted. The communication channels will be assumed to be under threat from an adversary. The communication channels will be used to transmit AES encrypted symbolic speech data, and return AES encrypted search results and transcriptions. This is the basis by which we will design suitable encryption.

4.1. Deterministic Encryption

Deterministic encryption always encrypts the same message to the same cipher text. The property preserved by deterministic encryption is equality, i.e. for any given two encryptions one can test if the underlying messages are equal by checking the given cipher texts. Due to this equality property, the encrypted database leaks a large amount of data even before the client searches. This means that if the server sees two or more equal cipher texts in the encrypted database, it knows that the corresponding encrypted documents contain a keyword in common. In addition, the server learns the frequency with which keywords appear which makes the encrypted database vulnerable to frequency analysis. Another issue is that since tokens are deterministic encryptions of the search terms, the server will always know whether the client

is repeating a search or not. A third issue occurs when the deterministic encryption scheme is based on a public-key scheme. In this case, all the deterministic encryptions (both in the encrypted database and in the tokens) are encrypted using the client's public key which is available to the server. The server can then mount a dictionary attack on the encrypted database by encrypting a list of possible keywords and comparing them to the ones found in the encrypted database and in the tokens. If it finds a match, then it knows the keyword. Hence the solution based on deterministic encryption supports fast search on encrypted data.

Searchable encryption based on identity based encryption (IBE) is secure in the traditional cipher text indistinguishability sense. The anonymity requirement informally states that cipher texts leak no information regarding the identity of the recipient, leading to the commonly desired keyword-privacy guarantees over cipher texts in searchable encryption. The standard notion of cipher text indistinguishability in IBE informally means that it is hard for computationally bounded adversaries to find two distinct keywords such that the trapdoors for the first keyword positively match the cipher texts of the second.

4.2. Ranked Searchable Encryption (RSE)

We will consider the client-server infrastructure by visualizing a scenario in which there are two parties, Alice (Client) and a cloud server. Alice intends to upload all her documents (encrypted speech files) $D = \{D_1, D_2, \ldots, D_i\}$ to the cloud server to enable remote access. The cloud server performs the searching of the relevant documents on behalf of Alice. In the scheme it is assumed that the cloud server acts in a known and designated manner but is equally also willing and curious to get hold of any information about the documents held with it. To prevent theft of any of the information Alice decides to encrypt all the documents. Once the documents are encrypted and outsourced she is challenged with the problem of searching on the encrypted documents. Whenever Alice decides to view a particular file she has to download all the documents from the cloud server and after decrypting all of them she can get hold of her required set of files. This creates unnecessary network traffic and post processing overhead. Alice decides to outsource the documents in such a way that she would only have to download the relevant and desired documents while keeping the security and privacy of the outsourced files intact. This requires a scheme to be developed that would facilitate performing textual searches over encrypted data.

Searching over encrypted documents is performed in three phases (Setup, Searching and Outcome). The first phase i.e. the Setup Phase, comprises the three steps Keyword Identification, Client Index Generation and Server Index Generation. In the first step Alice generates an exhaustive set of unique Keywords $W = \{W_1, W_2, \ldots, W_N\}$ from the set of documents D to be outsourced. Next Alice builds a client-side index table I_c. The I_c is stored with Alice and is never revealed to the cloud server. In the final step, Alice generates a secure ranked server-side index Is and outsources it to the cloud server along with the encrypted set of documents D. This involves the relevant frequencies of the keywords to be calculated and inserted into the index table.

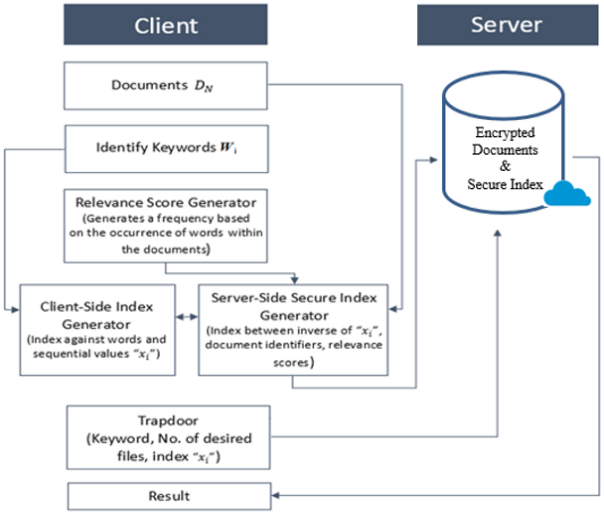

Fig. 3. RSE Flow of Events

In the Searching Phase, Alice generates a Trapdoor T_i for the particular keyword W_i she is willing to search. T_i is then transmitted to the cloud server to facilitate the search. In the Outcome Phase the cloud server returns the encrypted set of desired files to Alice in the ranked order. Fig. 3 shows the flow of events of the proposed RSE scheme where a client is interacting with a cloud server. It can be seen that all the tasks are performed by the client, whereas, the searching is done at the cloud server side.

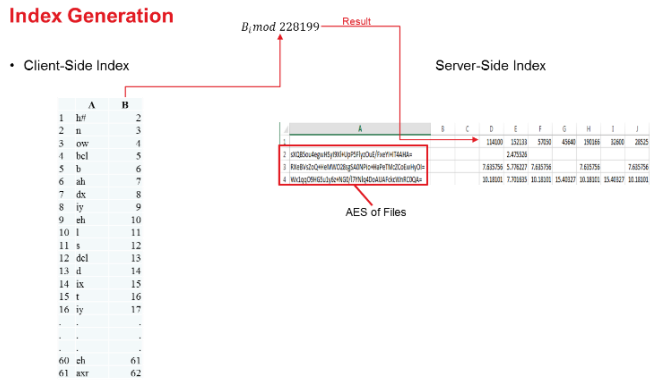

Fig. 4. Server-side index table generation

Pre-processing is done on the client side in three major steps, namely frequency computation, client-side index generation and server-side index generation. With frequency computation, the scheme computes the frequency of the

words appearing in each of the selected files. The next task is to generate the client-side index. The client side index table is a collection of all key words each assigned with a unique integer other than 0 and 1. If the total number of a set of keywords is, say, '*N*' in number, then a prime number '*p*' is chosen such that $p > N$. All integers that are assigned for the keywords are from the set $\{2, 3, \ldots, p\text{-}1\}$.

The client-side index table is simply just a key for the keywords, which in our case are phonetic symbols. The server-side index table keeps track of the distribution of the keywords throughout the documents. The server-side index table is a frequency table with three modifications: Firstly, the words are replaced by the integers which are the multiplicative inverse of their corresponding client-side index computed modulo prime *p*. For example, the client-side index of this is 2. The multiplicative inverse of 2 in modulo 228199 is 114100. So in the Server-side index table, the word 'this' is replaced by 114100. Secondly, the file names in the frequency table are replaced by the encrypted file names. For example, word.doc is replaced by AES (word.doc) as shown in the figure. Lastly, the frequencies are replaced by relevant scores which are computed using the following formula:

$$Score(Q, F_d) = \sum_{t \in Q} \frac{1}{|F_d|} \left(1 + \ln f_{d,t}\right) \ln\left(1 + \frac{N}{f_t}\right) \quad (1)$$

As is illustrated by Fig. 4, to search for a keyword, say the phoneme 'n', the client will compute E = (Decimal (AES ('n'))) mod 228199 and the trapdoor K = (Decimal (AES ('n')) * 3) mod 228199. Note that 3 is the client-side index of the phonetic symbol 'n'. On the server side, after receiving (K, E), K will be multiplied modulo 228199 with the integers occurring in the first row of server-side index table one by one unless product matches K. For example, since the multiplicative inverse of 3 in modulo 228199 is 152133, ($K \times$ 152133) mod 228199 = Decimal (AES ('n')) mod 228199 = E. Now the entries of the column corresponding to the integer 152133 are to be checked. The higher the score, the more relevant the corresponding file is with respect to the search. If the number of files in which the search is be performed is, say, 2, then the top two files according to the top two relevant scores for the keyword 'n' are 7.7 and 5.7 and the corresponding encrypted files are new.docx and Latest.doc. So the server will return the encrypted new.docx first and then Latest.doc.

4.2. Searching for Sequences of Phones

To improve the usability of the scheme we have recently extended it to enable the searching of strings of phonetic symbols. In effect a simple lexicon can reside on the client-side, and then the end-user can search using words, the lexicon can transform the search into strings of phones and then the encrypted search can be performed. The modification to the slight modification to the server-side index table. Basically the relevance score (Eq. 1) is replaced with a string of integers indicative of a hash chain. To implement the hash chain functionality, the server side index table has the various phonetic symbols coded into the column entries of the table. Take a randomly generated lambda-bit integer, say *r*, then the first column in the server side index table will be *r*, the next column will be $H(r)$, the third column will be $H^2(r)$ and in general the column corresponding to the *i*-th symbol will be $H^{i\text{-}1}(r)$, where $H()$ is a cryptographically strong keyed hash function (SHA-1 or SHA-2).

So to search for the encrypted audio on the cloud containing a particular word, say the word 'test' we look up the lexicon that says we need to search the encoded speech repository for the phonetic string 't eh s t'. Then all entries corresponding to the symbol 't' are masked with $H_{k_m}(t)$. Similarly, entries corresponding to other keywords are masked. Hence to search for the string 't eh s t' for each symbol s_i the client will compute:

$$\begin{aligned} k_i &= H(decimal(\text{AES}(\, s_i\,)) + c + H_{k_m}(s_i)) \\ k_i^{td} &= decimal\big(AES(\, s_i\,)\big) * c \bmod p \\ msk_i &= decimal(\text{AES}(\, s_i\,))^{-1} * H_{k_m}(s_i) \end{aligned} \quad (2)$$

where c is the client side index for the phonetic symbol under question. The search query is then of the form: $(\{k_1, k_1^{td}, msk_1\}, \{k_2, k_2^{td}, msk_2\}, \{k_3, k_3^{td}, msk_3\}, \{k_4, k_4^{td}, msk_4\})$. Using the RSE searchable encryption described previously, the server will detect the columns corresponding to 't', 'eh', 's', 't' and return the AES encrypted audio filename.

5. CONCLUSIONS

We have presented an outline of encrypted phonetic search of audio stored on the cloud. Whilst the complete system is still in development the rationale for how the system works has been presented along with specification and preliminary operation of many of the core modules. The rationale advocates that audio data is uploaded to the cloud and remains encrypted once it leaves the client-side. On the client-side the audio data is additionally encoded into symbols by a novel CNN based acoustic model. The encoded speech (phonetic symbolic strings) are then encrypted with AES and uploaded to the cloud. RSE provides the capability to perform phonetic searching of the encoded audio securely in the cloud realizing the speech recognition task. The inherent trapdoor security that RSE employs preserves the privacy of searches.

ACKNOWLEDGEMENTS

This research was supported by Innovate UK as part of the 'Privacy Preserving Speech Processing in the Cloud' project (Project No.: 102506).

REFERENCES

[1] Chollet, G. Cernocky, J. Constantinescu, A., et al.: 'Toward ALISP: A proposal for automatic language independent speech processing', Computational Models of Speech Pattern Processing, K. Ponting (ed.), NATO ASI Series, 1999, 169, pp 375-388.

[2] Curtmola, R. Garay, J. Kamara, S., et al.: 'Searchable symmetric encryption: Improved definitions and efficient constructions', Proc. 13th ACM Conf. Comput. Netw. Secur., 2006, pp. 79.

[3] Pathak, M.A., Raj, B., Rane, S., et al.: 'Privacy-preserving speech processing: cryptographic and string-match frameworks show promise', IEEE Signal Processing Magazine, 2013, 30(2), pp. 62-74.

[4] Hannun, A., Case, C., Casper, J., et al.: 'Deep speech: Scaling up end-to-end speech recognition', arXiv preprint arXiv:1412.5567, 2014.

[5] Graves, A., Fernández, S., Gomez, F., et al.: 'Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks', Proc. ACM 23rd Int. Conf. Machine Learning, 2006, pp. 369-376.

[6] 'DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CDROM' (1993) by Garofolo, J.S., Lamel, L.F., Fisher, W.M., et al.: https://catalog.ldc.upenn.edu/LDC93S1

[7] 'CAFFE Deep Learning Framework', http://caffe.berkeleyvision.org/

[8] 'NVIDIA DIGITS Interactive Deep Learning GPGPU Training System', https://developer.nvidia.com/digits